Enabling Efficient and Semantic-Aware Constraint Validation in Knowledge Graphs

Jin Ke*[0009−0001−8516−8894]

Technical University of Munich, Germany jin.ke@rub.de

Abstract. As the usage of RDF knowledge graphs (KGs) becomes more pervasive in practical applications, there is a burgeoning need for high-quality RDF data. The SHApes Constraint Language (SHACL) enables precise constraint expression on RDF graphs, ensuring data structure compliance. However, a SHACL validation that overlooks the crucial implicit information encoded in the ontology of the KG may result in unsound results. Semantic-aware SHACL validation addresses this by considering implicit information in RDF graphs, thus enabling thorough and accurate data validation. Current methods that incorporate entailment into SHACL validation often face efficiency challenges due to the resource-intensive nature of applying inference rules across entire datasets. In this doctoral work, we explore methods to enhance the efficiency of semantic-aware SHACL validation, presenting the problem statement, research questions, hypotheses. The paper concludes by our proposed method and sharing preliminary results from our research.

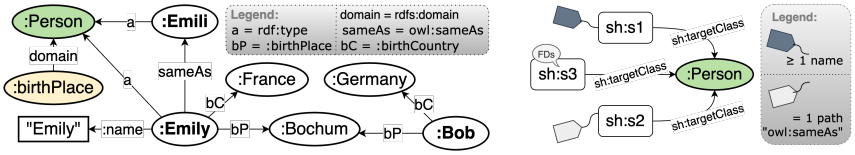
Keywords: SHACL · RDF · Data Quality · Reasoning · Constraints

1 Introduction

SHACL (SHApes Constraint Language) has been recommended by the W3C for the validation of knowledge graphs (KGs) modelled in RDF. As a language for expressing constraints on RDF graphs, SHACL enables the definition of RDF data structures, such as required properties, acceptable datatypes, and allowable values for specific properties. The process of SHACL validation acts as a safeguard, ensuring adherence to predefined schemas within RDF graphs, thereby enhancing the quality and interoperability of data [30], and paving the way for the development of advanced automated applications [21] [24].

Semantic-aware SHACL validation refers to the process of using the SHACL framework to validate RDF graphs under the context of semantics and inference. The incorporation of entailment enables SHACL engines to identify implicit relationships between entities, properties, and classes, thus achieving a thorough and accurate data validation. However, Corman et al. [7] have shown that the validation of RDF graphs against SHACL constraints, especially when involving recursion, is an NP-hard problem. Incorporating reasoning on top of SHACL validation adds another layer of complexity. Furthermore, SHACL lacks native

* Early Stage Ph.D.



(a) Data graph (b) Shapes graph

```
SELECT * WHERE { $e1 :bP ?p ; :bC ?c1 . ?e2 :bP ?p ; :bC ?c2 . FILTER (?c1 != ?c2) }
```

(c) SPARQL query for expressing the functional depency of `sh:s3`

Fig. 1. Motivating example. Shape `sh:s1` states every person should have at least one `:name`; and `sh:s2` states every person should have exactly one `owl:sameAs` property.

support for certain highly expressive constraint types, such as functional dependencies (FDs), making their implementation in SHACL less efficient. FDs state that some properties values are determined by the values of another set of properties. For instance, consider the example in Fig. 1, we verify for all `:Person` instances that `:birthPlace` → `:birthCountry` in `sh:s3`, i.e., if two people share the same birthplace, their birth countries should be the same. To achieve this, we have to implement SHACL validation using a SPARQL query in Fig. 1(c). Additionally, shapes graphs obtained with mining processes [27] often include numerous logically redundant constraints, leading to unnecessary repetitions in the validation process and consuming excessive computational resources. Besides, enabling entailment can lead to undesired behaviours. These issues often stem from the inherent reflexive nature of certain properties within the entailment framework, for instance, when following the *non-Unique Name Assumption (nUNA)* with `owl:sameAs` in KGs. To illustrate the effects of entailment, consider the example in Fig. 1. Without OWL reasoning, `:Bob` conforms to `sh:s1`, as it is not an instance of the `:Person` class. Yet, this is interpreted as a *false positive*, as when considering the domain of `:birthPlace`, it is entailed that this entity should be targeted by this shape. Therefore, with reasoning, the conclusion is that `:Bob` violates `sh:s1`. Moreover, reflexive properties can produce *spurious* validations with reasoning. Without reasoning, `:Bob` does not conform to `sh:s2`. This result is correct. However, with reasoning, due to the reflexivity of the `owl:sameAs` property, we entail the triple `(:Bob,owl:sameAs,:Bob)`, for which `:Bob` now conforms to `sh:s2`. This result can be considered *spurious*, as the conformance of the shape is due to redundant data entailed during the reasoning process and not due to the actual structure of the data graph. Besides, because of the nUNA, the violations found for `:Emily` are also reported for `:Emili`, which quickly inflates the validation reports with duplicate results. These show the importance of considering semantics to produce accurate results.

This doctoral dissertation is focused on tackling unresolved issues to enhance the efficiency of semantic-aware SHACL validation. Specifically, it delves into: *(i)* devising efficient strategies for integrating FDs within SHACL, *(ii)* reducing redundancy in shapes graphs to prevent repetitive validations, and *(iii)* investigating approaches to efficiently incorporate reasoning into SHACL validation.

2 State of the Art

First, as the problem of constraint validation has been extensively studied in other data models, we briefly revisit related work for relational databases and semi-structured sources. Then, we analyse rule-based validation approaches over KGs which is closely related to this thesis. Lastly, we position our work w.r.t. state-of-the-art SHACL validators over KGs that do not incorporate entailment.

Validation over other data models. The problem of constraint validation has been widely investigated in the context of data quality over the relational model and XML. The techniques for the relational model and databases focus on the discovery of different types of constraints, namely, functional dependencies (FDs), conditional FDs, pattern FDs, edit rules, denial constraints (e.g. [4, 8, 10, 26]). Most of these works present techniques to efficiently validate whether relations conform to the specified constraints. Other solutions present techniques to repair violations of these constraints (e.g. [2, 16, 20]). Although SHACL Core can express edit rule constraints well, it does not provide built-in constraint types for FDs constraints. Real-world scenarios require more complex shapes and SPARQL-based constraints (`sh:sparql`) to accurately implement and enforce FDs in RDF data using SHACL [15]. In comparison to these works, we do not present a validation or repair engine, but rather the strategies to apply targeted reasoning to the RDF graph and the shapes graph (with the constraints) to incorporate the semantics of ontologies using entailment regimes. Also, the aspect of reasoning in our work is not the same as the notion of "reasoning over functional dependencies" [8] also studied in databases. The latter typically involves applying Armstrong's axioms (reflexivity, augmentation, transitivity), computing the closure of attribute sets and analyzing normalization to ensure a well-designed database. Similarly to the literature on the relational model, several works have focused on the validation of semi-structured data models [17] and XML documents [3, 9] with different types of constraints, including keys, foreign keys, integrity constraints, functional dependencies, and embedded dependencies.

Rule-based validation over KGs. Several approaches support the inclusion of rules for SHACL validation. These rules can be defined in entailment regimes (e.g., RDFS, OWL) or in (domain-specific) inference rules. These approaches follow mainly two types of validation strategies, which extend the data graph or the shapes graph to incorporate the rules. Then, the actual validation process can be carried out with an off-the-shelf validator using the extended structures. One of these strategies relies on materializing the closure G^* of the data graph G for the set of rules; this strategy can be applied only in the presence of rules that do no introduce the existence of nodes that are not explicit in the original graph G. This strategy is implemented by pySHACL [29], which currently supports the entailment regimes RDFS [14] and OWL 2 RL [23]. The main drawbacks of this strategy are that G^* can be expensive to compute, and it contains a large number of redundant, inferred triples, exacerbating the performance of the validation process. A second strategy relies on rewriting the shapes graph $\mathcal{S}$ into a graph $\mathcal{S}'$, which extends the shapes to incorporate the information encoded in the inference rules. The works by Ahmetaj et al. [1] and Savković et al. [28]

provide formal shapes re-writings to incorporate the OWL 2 QL entailment regime into SHACL validation. These works provide guarantee the correcteness of SHACL validation with OWL 2 QL entailment. However, these works do not handle the semantics of the `owl:sameAs` predicate, as more expressive fragments of OWL 2 are required for this. Thus they cannot address the challenges posed by the nUNA in real-world scenarios. Validatrr [22] also applies shapes rewriting to support certain OWL features during SHACL validation. The main limitation of this work is that it does not provide theoretical guarantees, therefore, some OWL constructs may generate an exponential number of possible rewritings. To mitigate the issues of the two aforementioned strategies, a hybrid strategy can be applied where the inference and shapes rewriting is carried out in a targeted manner. For instance, the solution by Pareti et al. [24, 25] extends the data graph schema and rewrites the target queries and the constraints to incorporate inference rules. Another way of incorporating entailment regimes in validation is to decide SHACL shape containment. Leinberger et al. [19] explored how to determine SHACL shape containment using description logic reasoning, providing a formal approach to constraint validation in RDF data graphs. In this doctoral work, we will use these results to rewrite SHACL constraints into simpler, equivalent ones (without redundancy) that can be validated efficiently.

Validation without entailment over KGs. Other approaches [5, 29, 11] focus on the problem of efficient SHACL validation directly over the data graph, without considering the semantics of ontologies and entailment regimes. In contrast to these approaches, we propose solutions that enhance the shapes and data graph to remove redundancies and account for entailment regimes, before the actual validation is carried out. Therefore, our solution can be used in combination with any state-of-the-art SHACL validator.

3 Problem Statement and Contributions

SHACL validation requires two inputs, formally defined as follows [6, 7]: the *data graph* G is an RDF graph to be validated, and the *shapes graph* $\mathcal{S}$ which is a set of shapes defined as a triple $\langle s, targ_s, \varphi_s \rangle$, where s is a shape name, $targ_s$ is a (possibly empty) monadic query to a shape to retrieve the focus nodes, and φ_s is a constraint formula for s. The validation of $\mathcal{S}$ over G is denoted $[\![\mathcal{S}]\!]^G$ [7].

The semantic-aware SHACL validation checks the conformance of the data graph G following an entailment regime $\mathcal{E}$ to a shapes graph $\mathcal{S}$. In this case, the implicit information in G entailed by $\mathcal{E}$ can be taken into account in the validation process. We denote with $[\![\mathcal{S}]\!]^{G,\mathcal{E}}$ the validation of $\mathcal{S}$ over G under $\mathcal{E}$.

In practice, the computation of the validation result is executed by a validation strategy. We denote a SHACL validation strategy by $\mathbb{S}$ and the universe of strategies by $\mathfrak{S}$. To generate a validation result, a strategy $\mathbb{S}$ is executed on a given data graph G under the entailment regime $\mathcal{E}$ against the given shapes graph $\mathcal{S}$, which we denote by $\mathbb{S}(\mathcal{S}, G, \mathcal{E})$. Then, we use the execution runtime of the strategy $time\big(\mathbb{S}(\mathcal{S}, G, \mathcal{E})\big)$ to evaluate its efficiency. Therefore, we formalize our problem statement as the following optimization problem:

Problem Statement (Efficient SHACL Validation under Entailment): Given a data graph G, a shapes graph $\mathcal{S}$, and an entailment regime $\mathcal{E}$. The problem of efficient SHACL validation under entailment is defined as devising a validation strategy $\mathbb{S} \in \mathfrak{S}$ that minimizes the execution runtime while devising all violations to the shapes graph $\mathcal{S}$ over the data graph G under the entailment regime $\mathcal{E}$:

$$\arg\min_{\mathbb{S}\in\mathfrak{S}} \Big(time\big(\mathbb{S}(\mathcal{S}, G, \mathcal{E})\big)\Big), \text{ subject to } \mathbb{S}(\mathcal{S}, G, \mathcal{E}) = [\![\mathcal{S}]\!]^{G,\mathcal{E}}.$$

Contributions. To tackle this research problem, our novel contributions rely on the investigation of the following research questions:

RQ1 Expressiveness of SHACL Constraints:
- What is the expressivity of SHACL constraints w.r.t. constraints in other data models (e.g., edit rules or denial constraints)?
- How to validate expressive constraints (e.g., functional dependencies) efficiently in SHACL?

RQ2 Elimination of Redundancy in the Shapes Graph:
- How to remove duplicate constraints in the shapes graph when the validation does not consider reasoning?
- How to eliminate redundant constraints in the shapes graph when the validation includes reasoning?

RQ3 Efficient Reasoning over Data Graphs for Validation:
- How to reason over data graphs to improve the efficiency of SHACL validation with entailment?
- How to cope with the possible negative impact of nUNA on SHACL validation when enhancing RDF graphs through reasoning in OWL?

These research questions lead to our following hypothesis:

H1 Implementing highly expressive constraints such as FDs in SHACL typically requires the use of SPARQL expressions. Still, these constraints can be realized in SHACL without SPARQL by enhancing the data and shapes graphs.
H2 Shape rewriting can eliminate redundancy within the shapes graph while maintaining the expressiveness of the original one.
H3 Constraint-guided reasoning over the data graph allows for incorporating the semantics encoded in the ontology, while quickly producing a materialized graph that can be validated efficiently.

4 Research Methodology and Approach

The upcoming research and contributions aim to address the research questions posed by enhancing the data graphs and shapes graphs before validation. The methodology implemented for this doctoral research is structured as follows:

1. *Exploration of state-of-the-art techniques.* This involves a thorough review of existing literature pertinent to the problem at hand, focusing on constraints validation within the realms of Databases and Semantic Web.

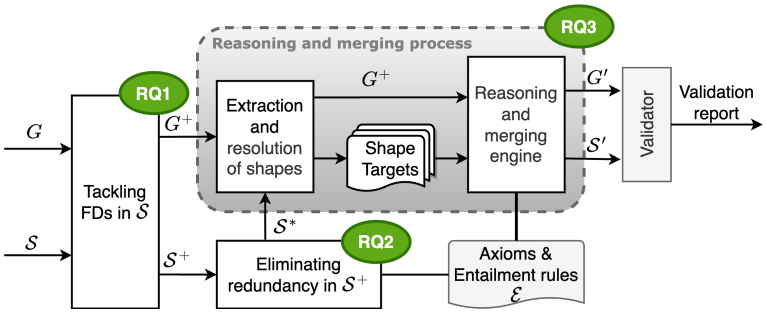


Fig. 2. Overview of our approach.

2. *Research questions and hypotheses formalization.* The research problem is clearly defined, leading to the formulation of specific research questions and hypotheses to guide the investigation.
3. *Devising the solutions.* Solutions are conceptualized to address the formulated hypotheses, focusing on the innovative aspects that these solutions bring to improve the efficiency of SHACL validation.
4. *Theoretical and empirical analyses.* The proposed solutions are analyzed rigorously in terms of their theoretical properties and performance in practice. This includes a detailed examination of the solutions' theoretical complexity to ensure their feasibility, and a comparative experimental evaluation against the state of the art to assess our solutions practical performance. The experimental steps are: *(i)* selecting the datasets, baseline methods and metrics to be used, *(ii)* developing experimental plans and evaluation strategies for the research questions, adapting existing benchmarks and evaluation metrics as needed, *(iii)* running our approaches and baseline methods according to the experimental plan, and *(iv)* analyzing experimental results to draw meaningful conclusions about the effectiveness and efficiency of the solutions.

We will propose corresponding approaches for each of our stated research questions. Fig. 2 depicts an overview of these approaches and labels the corresponding research questions. Regarding **H1**, we would like to study and compare the expressiveness of different types of constraints in different data models. Especially, we are interested in methods to implement FDs within SHACL. Since FDs target several nodes at the same time, it is only possible to express FDs using SPARQL-based constraints (`sh:sparql`) in SHACL, which are time consuming to execute. Instead, our approach will analyze the given FDs and generate G^+ where groups of target nodes are folded based on the attribute-value pairs to be validated in the FDs. Then, our approach obtains the rewritten shapes graph $\mathcal{S}^+$ by rewriting the SPARQL-based constraints as built-in SHACL constraints against the folded nodes. These techniques will allow for checking FDs efficiently.

We are inspired to study **H2** from works [19, 24, 25] and propose a rewriting solution for shapes graphs to eliminate the redundancy in them. First, we will present a method for detecting the redundant conditions in shapes graph with and without entailment. Then, we will devise an approach to implement shapes

rewriting to remove redundant conditions. In this way, our approach is able to understand the relationships between shapes by reasoning and to integrate these shape constraints into a more compact shapes graph $\mathcal{S}^*$ to be validated.

We address the hypothesis **H3** by presenting our approach, Re-SHACL. Instead of computing the closure G^* (under an entailment regime) of the data graph, Re-SHACL extracts relevant information from the shapes graph to identify the parts of the data graph for which the reasoning is applied to. To cope with the nUNA, Re-SHACL implements a merging mechanism, where semantically equivalent entities are consolidated into a single, unified representation in the data graph G'. Because of this merging, Re-SHACL rewrites the shapes graph to $\mathcal{S}'$ to replace the identifiers of the merged entities that occur in the shapes graph. The combination of the targeted reasoning and the entity merging, allows Re-SHACL to obtain concise, semantically enhanced data graphs, which can be efficiently evaluated using any SHACL validator without entailment.

5 Evaluation Plan

We plan to theoretically and experimentally evaluate the proposed methods. To provide theoretical guarantees for our methods, we will study their complexity and prove their correctness. Regarding the experimental evaluation, we will perform experiments and then analyze the results of different methods.

Benchmarks and Datasets. We will use synthetic data graphs (e.g., LUBM [13]) and real-world datasets (e.g., DBpedia [18] and Wikidata [31]). For the synthetic dataset, we have the flexibility to control the size of the generated dataset when synthesizing the database, which is helpful to fully analyze the effect of dataset dimensions on the experimental results. Real-world datasets contain real data obtained directly from real environments that do not follow the Unique Name Assumption (UNA). Therefore, the real-world datasets provide an important set up for assessing the studied methods. Regarding the shapes graphs, related works [11, 27] have published some shapes graphs designed for the dataset we have chosen. We will reuse and adapt these shapes graphs as much as possible.

Baselines. We will compare to approaches that support entailment regimes (e.g., pySHACL [29]), to measure the entailment regimes' impact on runtime and the number and type of violations that can be detected with reasoning. We will also use off-the-shelf validators (e.g., [5, 6, 11]) to show how our techniques can enhance the performance of state-of-the-art SHACL validators.

Metrics. To measure the performance of our methods, we will use several evaluation metrics, including the *execution time* of the approaches and *the number of violations* in the validation report. In the studies that handle the shapes graphs (i.e., for **H1** and **H2**) we also report on metrics about the shapes graph and the rewritten shapes graph, such as the *number of shapes* and *number of property constraints* in the graph. For the methods of enhancing data graphs (i.e., **H3**), we will also consider the metrics of *generation time* of the materialized graphs and their *size*. Studying these metrics helps us to better analyze the results of experiments, and to investigate which factors affect validation performance.

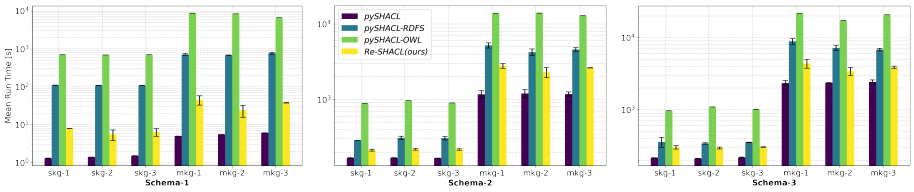
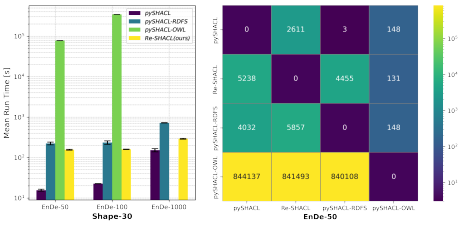


Fig. 3. Preliminary results over LUBM datasets using three shapes graphs.

6 Preliminary Results

We have delved into **H3** and propose Re-SHACL, an approach based on materialization and shape rewriting that considers the targets in the shapes to carry out targeted reasoning and merging to perform efficient validation with OWL LD entailment regime [12]. It follows a hybrid strategy with two main novelties: *(i)* it is tailored to entailment regimes and mitigates the undesired results obtained due to reflexive properties defined in RDFS and OWL and due to the nUNA followed in data graphs; and *(ii)* our shapes graph rewritings correspond to simple substitutions, which makes our solution efficient.



(a) (b)

Fig. 4. Preliminary results over DBpedia subsets with real-world shapes graphs: (a) Runtime (reasoning + validation). (b) Analysis of validation reports for the DBpedia subset EnDe-50. These values reflect how many violations appeared in the report R_y in the approach of the y-axis but not in the report R_x in the approach of x-axis.

We conducted experiments using six synthetic datasets LUBM generated by Figuera et al. [11] and three real-world subsets obtained from DBPedia. We use pySHACL as the validator for Re-SHACL, and compare to pySHACL with no/RDFS/OWL 2 RL entailment. Fig. 3 and Fig. 4(a) show the runtimes. Clearly the fastest approach is pySHACL without entailment. Re-SHACL exhibits a considerably lower runtime than pySHACL-RDFS in most configurations, despite that OWL LD reasoning (implemented by Re-SHACL) is more expressive than RDFS. In comparison to pySHACL-OWL, Re-SHACL is orders of magnitude faster in all datasets, especially in DBpedia. The LUBM results show that the runtime is affected by both the data and shapes graphs.

Fig. 4(b) represents the difference between the validation reports obtained with the different approaches. By analyzing the differences in the validation reports, we draw the following conclusions: *(i)* Using reasoning can effectively filter out numerous erroneous violations derived from the validation without reasoning, and discover more meaningful violations. *(ii)* pySHACL-OWL and pySHACL-RDFS exhibit a notable frequency of duplicated violations due to their inability to appropriately manage nUNA during validation, leading to redundancy within the entailment process. Conversely, Re-SHACL reports solely

fundamental violations because it supports OWL entailment with entity merging, thereby eliminating the repetition of violation reports. *(iii)* When dealing with reflexive properties, pySHACL-OWL may generate false positive violations due to OWL entailment causing every individual `e` in the data graph to imply the triple `(e,owl:sameAs,e)`, which conflicts with `sh:minCount` and `sh:maxCount` restrictions, leading to overlooked or incorrect validation outcomes.

Summarizing, our results show that Re-SHACL outperforms orders of magnitude existing approaches that support RDFS and OWL entailment.

7 Conclusions

This doctoral thesis focuses on exploring how to validate RDF data efficiently within the SHACL framework, considering entailment. We develop three hypotheses to tackle the pertinent research questions and outline an evaluation strategy to assess these hypotheses' viability. Initial findings for Hypothesis **H3** indicate that our introduced approach, Re-SHACL, employing shape-based inference and merging techniques, effectively supports entailment during validation. This preliminary result affirms our direction and gives us confidence in the ongoing research related to **H1** and **H2**.

Future work for H3 involves enhancing the method to support OWL 2 RL and advanced features in SHACL. Our subsequent focus will be on examining the other two hypotheses, **H1** and **H2**. Finally, we will discuss how to integrate the solutions derived from all three hypotheses to optimize validation efficiency.

Acknowledgement. I would like to thank my supervisor Prof. Dr. Maribel Acosta for her strong support and insightful feedback.

References

1. S. Ahmetaj, M. Ortiz, A. Oudshoorn, and M. Šimkus. Reconciling shacl and ontologies: semantics and validation via rewriting. In *ECAI*, pages 27–35. 2023.
2. A. Bronselaer and M. Acosta. Parker: Data fusion through consistent repairs using edit rules under partial keys. *Inf. Fusion*, 100:101942, 2023.
3. P. Buneman, W. Fan, J. Simeon, and S. Weinstein. Constraints for semistructured data and xml. *ACM Sigmod Record*, 30(1):47–54, 2001.
4. X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *Proc. VLDB Endow.*, 6(13):1498–1509, 2013.
5. J. Corman, F. Florenzano, J. L. Reutter, and O. Savkovic. Shacl2sparql: Validating a sparql endpoint against recursive shacl constraints. In *The Semantic Web-ISWC*, volume 2456 of *CEUR Workshop Proceedings*, pages 165–168, 2019.
6. J. Corman, F. Florenzano, J. L. Reutter, and O. Savkovic. Validating shacl constraints over a sparql endpoint. In *The Semantic Web-ISWC*, volume 11778, pages 145–163. Springer, 2019.
7. J. Corman, J. L. Reutter, and O. Savkovic. Semantics and validation of recursive SHACL. In *ISWC*, volume 11136, pages 318–336. Springer, 2018.
8. W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM TODS*, 33(2):1–48, 2008.

9. W. Fan and J. Siméon. Integrity constraints for xml. In *Proceedings of the ACM SIGMOD-SIGACT-SIGAI Symposium on PODS*, pages 23–34. ACM, 2000.
10. I. P. Fellegi and D. Holt. A systematic approach to automatic edit and imputation. *Journal of the American Statistical association*, 71(353):17–35, 1976.
11. M. Figuera, P. D. Rohde, and M. Vidal. Trav-shacl: Efficiently validating networks of SHACL constraints. In *WWW*, pages 3337–3348. ACM / IW3C2, 2021.
12. B. Glimm, A. Hogan, M. Krötzsch, and A. Polleres. OWL: yet to arrive on the web of data? In *LDOW*, volume 937 of *CEUR Workshop Proceedings*, 2012.
13. Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2-3):158–182, 2005.
14. P. J. Hayes and P. F. Patel-Schneider. *RDF 1.1 Semantics*. URL: `https://www.w3.org/TR/rdf11-mt/\#rdf-interpretations`, 2014. W3C Recommendation.
15. H. Knublauch and D. Kontokostas. *Shapes Constraint Language (SHACL)*. URL: `https://www.w3.org/TR/shacl-js/`, 2017. W3C Recommendation.
16. S. Kolahi and L. V. Lakshmanan. On approximating optimum repairs for functional dependency violations. In *ICDT*, volume 361, pages 53–62. ACM, 2009.
17. F. Korn, B. Saha, D. Srivastava, and S. Ying. On repairing structural problems in semi-structured data. *Proc. VLDB Endow.*, 6(9):601–612, 2013.
18. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.
19. M. Leinberger, P. Seifer, T. Rienstra, R. Lämmel, and S. Staab. Deciding SHACL shape containment through description logics reasoning. In *The Semantic Web-ISWC*, volume 12506 of *Lecture Notes in Computer Science*, pages 366–383, 2020.
20. E. Livshits, B. Kimelfeld, and S. Roy. Computing optimal repairs for functional dependencies. *ACM TODS*, 45(1):1–46, 2020.
21. J. Mäkelburg, C. John, and M. Acosta. Automation of electronic invoice validation using knowledge graph technologies. In *Proceedings of ESWC*, 2024. (To appear).
22. B. D. Meester, P. Heyvaert, D. Arndt, A. Dimou, and R. Verborgh. RDF graph validation using rule-based reasoning. *Semantic Web*, 12(1):117–142, 2021.
23. B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. *OWL 2 Web Ontology Language Profiles*, 2012. W3C Recommendation.
24. P. Pareti and G. Konstantinidis. A review of SHACL: from data validation to schema reasoning for RDF graphs. In *Reasoning Web*, pages 115–144, 2021.
25. P. Pareti, G. Konstantinidis, T. J. Norman, and M. Şensoy. Shacl constraints with inference rules. In *The Semantic Web-ISWC*, volume 11778 of *Lecture Notes in Computer Science*, pages 539–557. Springer, 2019.
26. A. A. Qahtan, N. Tang, M. Ouzzani, Y. Cao, and M. Stonebraker. Pattern functional dependencies for data cleaning. *Proc. VLDB Endow.*, 13(5):684–697, 2020.
27. K. Rabbani, M. Lissandrini, and K. Hose. Extraction of validating shapes from very large knowledge graphs. *Proc. VLDB Endow.*, 16(5):1023–1032, 2023.
28. O. Savković, E. Kharlamov, and S. Lamparter. Validation of shacl constraints over kgs with owl 2 ql ontologies via rewriting. In *The Semantic Web-ESWC*, volume 11503 of *Lecture Notes in Computer Science*, pages 314–329. Springer, 2019.
29. A. Sommer, N. Car, and J. Yu. *pySHACL*. URL: `https://pypi.org/project/pyshacl/0.9.5/`, 2018. A Python validator for SHACL.
30. S. Stolk and K. McGlinn. Validation of ifcowl datasets using SHACL. In *LDAC*, volume 2636 of *CEUR Workshop Proceedings*, pages 91–104. CEUR-WS.org, 2020.
31. D. Vrandecic and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.